Übung zur Vorlesung

Automatisierte Logik und Programmierung

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2013/14

Blatt 3 — Abgabetermin: 22.11.2013

Das dritte Übungsblatt soll dazu dienen, sich mit Fragen der Korrektheit von Kalkülen auseinanderzusetzen und Erfahrungen mit Gleichheitsbeweisen und dem λ–Kalkül zu sammeln.
Wir werden mögliche Lösungen zu Beginn der Veranstaltung am 22.11.2013 besprechen

---

Aufgabe 3.1 (Korrektheit der Refinement Logik)

Zeigen Sie, daß die prädikatenlogische Refinement Logik korrekt ist.

3.1–a Beweisen Sie, daß die Regeln `impliesR`, `impliesL`, `orL` und `exL` korrekt sind, und nehmen Sie im folgenden an, daß alle anderen Regeln ebenfalls als korrekt bewiesen sind.

3.1–b Beweisen Sie durch Induktion über Beweisbäume: "Wenn alle Regeln eines Kalküls korrekt sind, dann haben alle Theoreme gültige Initialsequenzen als Wurzeln"

3.1–c Beweisen Sie: "Eine Formel C ist gültig, wenn die Initialsequenz $\vdash C$ gültig ist"

3.1–d Zeigen Sie: "Eine Formel C ist gültig, wenn $\vdash C$ in Refinement Logik beweisbar ist"

Aufgabe 3.2 (Gleichheit)

Beweisen Sie folgende Formeln mithilfe der Refinement Logik einschließlich der Gleichheitsregeln:

3.2–a $(\forall n)(\forall m)(m{=}n \;\Rightarrow\; s(n){=}s(m))$

3.2–b $(g(g(g(a))){=}a \wedge g(g(g(g(a)))){=}a) \;\Rightarrow\; g(a){=}a$

3.2–c $(plus(n,0){=}n \wedge (\forall n)(\forall m)(plus(n,s(m)){=}s(plus(n,m)))) \;\Rightarrow\; plus(s(0),s(s(0))){=}s(s(s(0)))$

Aufgabe 3.3 (λ–Kalkül: Boolesche Algebra)

Definieren Sie mithilfe von T, F und $\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t$ die folgenden Booleschen Operatoren:

3.3–a "and", die logische Konjunktion

3.3–b "or", die logische Disjunktion

3.3–c "neg", die logische Negation

3.3–d "imp", die logische Implikation

und geben Sie diese anschließend als reine λ-Terme (ohne Verwendung abkürzender Definitionen) an. Überlegen Sie sich, wie man die Korrektheit Ihrer Definitionen formal nachweisen könnte und skizzieren Sie für jeden Ihrer Operatoren einen Korrektheitsbeweis.

Aufgabe 3.4 (λ–Kalkül: Ganzzahlfunktionen)

3.4–a Geben Sie einen λ-Term subtract an mit der Eigenschaft $\mathsf{subtract}\ \overline{m}\ \overline{n} = \overline{m-n}$

3.4–b Beschreiben Sie einen λ-Term less_or_equal mit $\mathsf{less\_or\_equal}\ \overline{m}\ \overline{n} = \begin{cases} \mathsf{T} & \text{falls } m{\leq}n \\ \mathsf{F} & \text{sonst} \end{cases}$

3.4–c Geben Sie einen λ-Term max an mit der Eigenschaft $\mathsf{max}\ \overline{m}\ \overline{n} = \begin{cases} \overline{n} & \text{falls } m{\leq}n \\ \overline{m} & \text{sonst} \end{cases}$

Skizzieren Sie jeweils einen Korrektheitsbeweis.

Aufgabe 3.5 (λ–Kalkül: Korrektheit von Repräsentationen)

Die Darstellung der natürlichen Zahlen wurde mit Hilfe der *Church Numerals* ($\overline{n} \equiv \lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^n\ \mathtt{x}$) beschrieben. Beweisen Sie die Korrektheit der folgenden Operationen

3.5–a $\mathsf{PRs}[b,h] \equiv \lambda n.n\ h\ b$

Zeigen Sie, daß für $f \equiv \mathsf{PRs}[b,h]$ gilt: $f\ \overline{0} = b$ und $f\ (s\ n) = h\ (f\ n)$

3.5–b $\mathsf{p} \equiv \lambda \mathtt{n}.\ (\mathtt{n}\ (\lambda \mathtt{f\_x}.\ \langle \mathtt{s}, \mathsf{match}\ \mathtt{f\_x}\ \mathsf{with}\ \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \ \mathtt{x}\rangle)\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle).2$ (sehr aufwendig)

Zeigen Sie, daß p die Vorgängerfunktion repräsentiert, d.h. für alle n gilt $\mathsf{p}\ \overline{n} = \overline{n-1}$.

Lösung 3.1

3.1–a Beweisen Sie, daß die Regeln `impliesR`, `impliesL`, `orL` und `exL` korrekt sind, und nehmen Sie im folgenden an, daß alle anderen Regeln ebenfalls als korrekt bewiesen sind.

Die Korrektheit einer Regel $r = (dec, val)$ war wie folgt definiert: *Sind* $S_i = H_i \vdash C_i$ *Ergebnis der Anwendung von* dec *auf* $S = H \vdash C$ *und* c_i *Evidenzterme für* S_i*, dann ist* $val(S, (S_i, c_i))$ *Evidenzterm für* S. Die konkreten Beweise folgen den Begründungen für die Konstruktion der Evidenzterme in den Regeln.

$$\begin{array}{l} H \vdash A \Rightarrow B \quad \lfloor \text{ext } \lambda a.b \rfloor \\ \quad H, a{:}A \vdash B \quad \lfloor \text{ext } b \rfloor \end{array} \quad \texttt{impliesR}$$

Sei b Evidenzterm für die Sequenz $S_1 = H, a{:}A \vdash B$. Dann ist b Evidenz für die Formel B, wenn alle Variablen von H mit Evidenzen für die deklarierten Formeln in H und a mit Evidenz für A instantiiert werden. Dann ist auch $\lambda a.b = val(H \vdash A \Rightarrow B, (S_1, b)$ Evidenz für die Formel $A \Rightarrow B$, wenn alle Variablen von H mit Evidenzen für die deklarierten Formeln in H instantiiert werden, also ein Evidenzterm für die Sequenz $S = H \vdash A \Rightarrow B$.

$$\begin{array}{l} H, f{:}A \Rightarrow B, H' \vdash C \quad \lfloor \text{ext } c[f(a)/b] \rfloor \\ \quad H, f{:}A \Rightarrow B, H' \vdash A \quad \lfloor \text{ext } a \rfloor \\ \quad H, b{:}B, H' \vdash C \quad \lfloor \text{ext } c \rfloor \end{array} \quad \texttt{impliesL}$$

Der Beweis folgt analog der informalen Begründung für die Konstruktion des Evidenzterms.

$$\begin{array}{l} H, x{:}A \vee B, H' \vdash C \quad \lfloor \text{ext case } x \text{ of } \texttt{inl}(a) \rightarrow c_1 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mid \texttt{inr}(b) \rightarrow c_2 \rfloor \\ \quad H, a{:}A, H' \vdash C \quad \lfloor \text{ext } c_1 \rfloor \\ \quad H, b{:}B, H' \vdash C \quad \lfloor \text{ext } c_2 \rfloor \end{array} \quad \texttt{orL}$$

Der Beweis folgt analog der informalen Begründung für die Konstruktion des Evidenzterms.

$$\begin{array}{l} H, z{:}(\exists x)B, H' \vdash C \quad \lfloor \text{ext } c[z.1, z.2/x', b] \rfloor \\ \quad H, x'{:}\mathbb{U}, b{:}B[x'/x], H' \vdash C \quad \lfloor \text{ext } c \rfloor \end{array} \quad \texttt{exL}$$

Der Beweis folgt analog der informalen Begründung für die Konstruktion des Evidenzterms.

3.1–b Beweisen Sie durch Induktion über Beweisbäume: "Wenn alle Regeln eines Kalküls korrekt sind, dann haben alle Theoreme gültige Initialsequenzen als Wurzeln"

Induktion über die Tiefe des Beweisbaums zeigt, daß alle Sequenzen im Baum gültig sind

Blätter haben gültige Sequenzen, da die Validierung aus dem Nichts eine Evidenz hierfür konstruiert.

Bei inneren Knoten konstruiert die Validierung aus Evidenztermen der Teilsequenzen, die nach Induktionsannahme existieren, einen Evidenzterm für die Hauptsequenz.

Damit ist die Initialsequenz an der Wurzel ebenfalls gültig.

3.1–c Beweisen Sie: "Eine Formel C ist gültig, wenn die Initialsequenz $\vdash C$ gültig ist"

Wenn die Initialsequenz $\vdash C$ gültig ist, dann gibt es per Definition einen Evidenzterm für $\vdash C$.

Der Evidenzterm für eine Sequenz $x_1{:}A_1, .., x_n{:}A_n \vdash C$ war definiert als Term e mit freien Variablen aus den x_i, so daß e Evidenz für die Formel C ist, wenn alle x_i mit Evidenz für A_i instantiiert werden

Da eine Initialsequenz keine Hypothesen enthält ist der Evidenzterm für $\vdash C$ ein Term e ohne freie Variablen, der Evidenz für die Formel C ist. Per Definition ist C damit gültig.

3.1–d Zeigen Sie: "Eine Formel C ist gültig, wenn $\vdash C$ in Refinement Logik beweisbar ist"

Dies faßt das obige zusammen. Da alle Regeln der Refinement Logik korrekt sind, hat jedes Theorem der Refinement Logik gültige Initialsequenzen als Wurzeln. Wenn also $\vdash C$ in Refinement Logik beweisbar, dann ist $\vdash C$ und damit auch die Formel C gültig.

Lösung 3.2

Ziel dieser Aufgabe ist es, außer der vertieften Auseinandersetzung mit dem prädikatenlogischen Sequenzenkalkül, ein gewisses Gefühl für den Umgang mit Gleichheit zu bekommen. Die dazu ausgewählten Beispiele dürften dabei von ausreichendem Umfang sein...

Die etwas eigenartige Hypothesennumerierung in den folgenden NuPRL–Ableitungen beruht darauf, daß die Typisierungshypothesen für Bereiche, Prädikate und Funktionen der Übersicht halber weggelassen wurden:

3.2–a $\vdash \quad \forall n,m:\mathbb{N}.\ \ m{=}n \Rightarrow s(n){=}s(m)$:

```
⊢ ∀n:ℕ.  ∀m:ℕ.  m = n ⇒ s n = s m                                  BY allI THEN allI
\
 3.  n:  ℕ
 4.  m:  ℕ
  ⊢ m = n ⇒ s n = s m                                                BY impI
 \
  5.  m = n
  ⊢ s n = s m                                          BY substitution 'm = n'
  |\
  | ⊢ m = n                                                        BY hypothesis 5
   \
    ⊢ s n = s n                                                     BY reflexivity
```

3.2–b $\vdash \quad \forall a:\mathbb{O}.\ (g(g(g(a))){=}a \wedge g(g(g(g(a))))){=}a) \Rightarrow g(a){=}a$:

```
⊢ ∀a:O. g (g (g a)) = a ∧ g (g (g (g a))) = a ⇒ g a = a                BY allI
\
 3.  a:  O ⊢ g (g (g a)) = a ∧ g (g (g (g a))) = a ⇒ g a = a            BY impI
 \
  4.  g (g (g a)) = a ∧ g (g (g (g a))) = a ⊢ g a = a                  BY andE 4
  \
   4.  g (g (g a)) = a
   5.  g (g (g (g a))) = a
    ⊢ g a = a                                  BY substitution 'a = g (g (g (g a)))'
   |\
   | ⊢ a = g (g (g (g a)))                                            BY symmetry
   | \
   |  ⊢ g (g (g (g a))) = a                                          BY hypothesis 5
   \
    ⊢ g (g (g (g (g a)))) = g (g (g (g a)))
    |                                  BY substitution 'g (g (g (g (g a)))) = a'
    |\
    | ⊢ g (g (g (g (g a)))) = a                  BY substitution 'g (g (g a)) = a'
    | |\
    | | ⊢ g (g (g a)) = a                                             BY hypothesis 4
    |  \
    |   ⊢ g (g (g a)) = a                                            BY hypothesis 4
    \
     ⊢ a = g (g (g (g a)))                                            BY symmetry
     \
      ⊢ g (g (g (g a))) = a                                          BY hypothesis 5
```

3.2–c ⊢ ∀0:ℕ. (∀n,m:ℕ. plus(n,s(m))=s(plus(n,m)) ∧ plus(n,0)=n)
⇒ plus(s(s(0)),s(s(s(0)))) = s(s(s(s(0)))):

```
⊢ ∀0:ℕ
|   (∀n:ℕ. ∀m:ℕ. plus n (s m) = s (plus n m) ∧ plus n 0 = n)
|   ⇒ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))                    BY allI
\
 4. 0: ℕ
  ⊢ (∀n:ℕ. ∀m:ℕ. plus n (s m) = s (plus n m) ∧ plus n 0 = n)
 |  ⇒ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))                    BY impI
 \
  5. ∀n:ℕ. ∀m:ℕ. plus n (s m) = s (plus n m) ∧ plus n 0 = n
   ⊢ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))          BY allE 5 's(s 0)'
   \
    6. ∀m:ℕ. plus (s(s 0)) (s m) = s (plus (s(s 0)) m) ∧ plus (s(s 0)) 0 = s(s 0)
    ⊢ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))          BY allE 6 's(s 0)'
    \
     7. plus (s(s 0)) (s (s(s 0))) = s (plus (s(s 0)) (s(s 0)))
              ∧ plus (s(s 0)) 0 = s(s 0)
      ⊢ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))                  BY andE 7
      \
       7. plus (s(s 0)) (s (s(s 0))) = s (plus (s(s 0)) (s(s 0)))
       8. plus (s(s 0)) 0 = s(s 0)
        ⊢ plus (s(s 0)) (s (s(s 0))) = s (s (s (s(s 0))))
        |          BY substitution 'plus (s(s 0)) (s (s(s 0))) = s (plus (s(s 0)) (s(s 0)))'
        |\
        | ⊢ plus (s(s 0)) (s (s(s 0))) = s (plus (s(s 0)) (s(s 0)))      BY hypothesis 7
        \
         ⊢ s (plus (s(s 0)) (s(s 0))) = s (s (s (s(s 0))))               BY allE 6 's 0'
         \
          9. plus (s(s 0)) (s(s 0)) = s (plus (s(s 0)) (s 0)) ∧ plus (s(s 0)) 0 = s(s 0)
           ⊢ s (plus (s(s 0)) (s(s 0))) = s (s (s (s(s 0))))                BY andE 9
           \
            9. plus (s(s 0)) (s(s 0)) = s (plus (s(s 0)) (s 0))
            10. plus (s(s 0)) 0 = s(s 0)
             ⊢ s (plus (s(s 0)) (s(s 0))) = s (s (s (s(s 0))))
             |          BY substitution 'plus (s(s 0)) (s(s 0)) = s (plus (s(s 0)) (s 0))'
             |\
             | ⊢ plus (s(s 0)) (s(s 0)) = s (plus (s(s 0)) (s 0))      BY hypothesis 9
             \
              ⊢ s (s (plus (s(s 0)) (s 0))) = s (s (s (s(s 0))))          BY allE 6 '0'
              \
               11. plus (s(s 0)) (s 0) = s (plus (s(s 0)) 0) ∧ plus (s(s 0)) 0 = s(s 0)
                ⊢ s (s (plus (s(s 0)) (s 0))) = s (s (s (s(s 0))))           BY andE 11
                \
                 11. plus (s(s 0)) (s 0) = s (plus (s(s 0)) 0)
                 12. plus (s(s 0)) 0 = s(s 0)
                  ⊢ s (s (plus (s(s 0)) (s 0))) = s (s (s (s(s 0))))
                  |          BY substitution 'plus (s(s 0)) (s 0) = s (plus (s (s 0)
                  |\
                  | ⊢ plus (s(s 0)) (s 0) = s (plus (s(s 0)) 0)          BY hypothesis 11
                  \
                   ⊢ s (s (s (plus (s(s 0)) 0))) = s (s (s (s(s 0))))
                   |                          BY substitution 'plus (s(s 0)) 0 = s(s 0)'
                   |\
                   | ⊢ plus (s(s 0)) 0 = s(s 0)                          BY hypothesis 12
                   \
                    ⊢ s (s (s (s(s 0)))) = s (s (s (s(s 0))))             BY reflexivity
```

Lösung 3.3

Die Lösung stammt aus einem alten Übungsblatt und verwendet eventuell veraltete Notation.

3.3–a "and", die logische Konjunktion:

Eine kurze Überlegung führt zu folgender Fallunterscheidung:

1. Ist A wahr, so ist $A \wedge B$ wahr, genau dann wenn B wahr ist.
2. Ist A falsch, so ist $A \wedge B$ sowieso falsch.

Damit können wir and wie folgt definieren:

$$\begin{aligned} \mathsf{and} &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{if}\ \mathtt{a}\ \mathsf{then}\ \mathtt{b}\ \mathsf{else}\ \mathsf{F} \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{cond}(\mathtt{a};\mathtt{b};\mathsf{F}) \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\ \mathtt{a}\ \mathtt{b}\ \mathsf{F} \end{aligned}$$

Wir zeigen zunächst den Beweis im Sequenzenkalkül für die Gleichheit von λ–Termen mit dem Eingabefall $\gg$T T$\ll$:

```
⊢ (λa,b.a b (λx,y.y)) (λx,y.x)(λx,y.x) = λx,y.x
                                        BY transitivity '(λb.(λx,y.x) b (λx,y.y))(λx,y.x)'
|\
| ⊢ (λa,b.a b (λx,y.y))(λx,y.x)(λx,y.x) = (λb.(λx,y.x) b (λx,y.y))(λx,y.x)
|                                                                          BY applyEq
| |\
| | ⊢ (λa,b.a b (λx,y.y))(λx,y.x) = λb.(λx,y.x) b (λx,y.y)                 BY reduction
| | \
| |  ⊢ λb.(λx,y.x) b (λx,y.y) = λb.(λx,y.x) b (λx,y.y)                     BY reflexivity
| |  \
| |   ⊢ λx,y.x = λx,y.x                                                    BY reflexivity
\
 ⊢ (λb.(λx,y.x) b (λx,y.y))(λx,y.x) = λx,y.x                               BY reduction
 \
  ⊢ (λx,y.x)(λx,y.x)(λx,y.y) = λx,y.x              BY transitivity '(λy,x,y.x)(λx,y.y)'
  |\
  | ⊢ (λx,y.x)(λx,y.x)(λx,y.y) = (λy,x,y.x)(λx,y.y)                         BY applyEq
  | |\
  | | ⊢ (λx,y.x)(λx,y.x) = λy,x,y.x                                        BY reduction
  | | \
  | |  ⊢ λy,x,y.x = λy,x,y.x                                              BY reflexivity
  | |  \
  | |   ⊢ λx,y.y = λx,y.y                                                 BY reflexivity
  |  \
  |   ⊢ (λy,x,y.x)(λx,y.y) = λx,y.x                                        BY reduction
  |   \
  |    ⊢ λx,y.x = λx,y.x                                                  BY reflexivity
```

Als nächstes folgt der Fall ≫F T≪:

```
⊢ (λa,b.a b (λx,y.y))(λx,y.y)(λx,y.x) = λx,y.y
                                          BY transitivity '(λb.(λx,y.y) b (λx,y.y))(λx,y.x)'
|\
| ⊢ (λa,b.a b (λx,y.y))(λx,y.y)(λx,y.x) = (λb.(λx,y.y) b (λx,y.y))(λx,y.x)
|                                                                          BY applyEq
| |\
| | ⊢ (λa,b.a b (λx,y.y))(λx,y.y) = λb.(λx,y.y) b (λx,y.y)                BY reduction
| |  \
| |   ⊢ λb.(λx,y.y) b (λx,y.y) = λb.(λx,y.y) b (λx,y.y)                  BY reflexivity
|  \
|   ⊢ λx,y.x = λx,y.x                                                    BY reflexivity
 \
  ⊢ (λb.(λx,y.y) b (λx,y.y))(λx,y.x) = λx,y.y                            BY reduction
  \
   ⊢ (λx,y.y)(λx,y.x)(λx,y.y) = λx,y.y                BY transitivity '(λy.y)(λx,y.y)'
   |\
   | ⊢ (λx,y.y)(λx,y.x)(λx,y.y) = (λy.y)(λx,y.y)                          BY applyEq
   | |\
   | | ⊢ (λx,y.y)(λx,y.x) = λy.y|                                         BY reduction
   | |  \
   | |   ⊢ λy.y = λy.y                                                   BY reflexivity
   |  \
   |   ⊢ λx,y.y = λx,y.y                                                BY reflexivity
    \
     ⊢ (λy.y)(λx,y.y) = λx,y.y                                           BY reduction
      \
       ⊢ λx,y.y = λx,y.y                                                 BY reflexivity
```

Alle anderen Fälle sind derart analog, daß wir sie uns ersparen wollen...

3.3–b "or", die logische Disjunktion:

Analog zu den obigen Überlegungen kommen wir zu folgender Definition für or:

$$\begin{aligned} \mathsf{or} &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{if}\ \mathtt{a}\ \mathsf{then}\ \mathsf{T}\ \mathsf{else}\ \mathtt{b} \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{cond}(\mathtt{a};\mathsf{T};\mathtt{b}) \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\ \mathtt{a}\ \mathsf{T}\ \mathtt{b} \end{aligned}$$

Auch hier gilt wiederum, daß die Sequenzenbeweise extrem analog zu den oben vorgeführten sind, so daß wir sie geflissentlich weglassen wollen.

3.3–c "neg", die logische Negation:

$$\begin{aligned} \mathsf{neg} &\equiv \lambda \mathtt{a}.\mathsf{if}\ \mathtt{a}\ \mathsf{then}\ \mathsf{F}\ \mathsf{else}\ \mathsf{T} \\ &\equiv \lambda \mathtt{a}.\mathsf{cond}(\mathtt{a};\mathsf{F};\mathsf{T}) \\ &\equiv \lambda \mathtt{a}.\ \mathtt{a}\ \mathsf{F}\ \mathsf{T} \end{aligned}$$

Sequenzenbeweise s. o.

3.3–d "imp", die logische Implikation:

$$\begin{aligned} \mathsf{imp} &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{if}\ \mathtt{a}\ \mathsf{then}\ \mathtt{b}\ \mathsf{else}\ \mathsf{T} \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\mathsf{cond}(\mathtt{a};\mathtt{b};\mathsf{T}) \\ &\equiv \lambda \mathtt{a}.\lambda \mathtt{b}.\ \mathtt{a}\ \mathtt{b}\ \mathsf{T} \end{aligned}$$

Sequenzenbeweise s. o.

Lösung 3.4

Die Lösung stammt aus einem alten Übungsblatt und verwendet eventuell veraltete Notation.

3.4–a subtract: Definitionsgemäß unterscheiden wir zwei Fälle für den zweiten Eingabeparameter $\overline{y}$: $y = 0$ oder $y > 0$. Im ersten Fall geben wir den ersten Eingabeparameter ($\overline{x}$) zurück, im zweiten gehen wir davon aus, daß wir für den Vorgänger von $\overline{y}$ das Ergebnis kennen und wenden auf dieses die Vorgängerfunktion an. Dies klingt nun verdächtig nach einem Fall für den einfachen Rekursionsoperator "PRs[$base$,h]":

Für "$base$" muß somit der erste Parameter "$\overline{x}$" herhalten und für "h" demzufolge "p". Das Ganze sieht dann so aus:

$$\mathsf{subtract} \equiv \lambda \mathtt{x}.\,\lambda \mathtt{y}.\,\mathsf{PRs}[\mathtt{x},\mathsf{p}]\ \mathtt{y}$$

Wem's Spaß macht, der kann ja mal den "nackten" λ–Term dazu hinschreiben.

Subtraktion von n ist n-fache Anwendung der Vorgängerfunktion p. *Da das Church Numeral $\overline{n}$ als die n-fache Anwendung einer Funktion auf ein Argument codiert ist, kann der Term direkt angegeben werden als* sub $\equiv \lambda m.\lambda n.\ n\ \mathsf{p}\ m$ *(Achtung: implizite Linksklammerung)*

Es ist

$\mathsf{sub}\ \overline{m}\ \overline{n}\ \equiv (\lambda m.\lambda n.\ n\ \mathsf{p}\ m)\ \overline{m}\ \overline{n}$
$\longrightarrow \overline{n}\ \mathsf{p}\ \overline{m}\ \equiv (\lambda f.\lambda x.f^n\ x)\ \mathsf{p}\ \overline{m}$
$\longrightarrow \mathsf{p}^n\ \overline{m}$
$\overset{*}{\longrightarrow} \overline{m-n}$, *weil* $\mathsf{p}\ \overline{m} = \overline{m-1}$ *gilt.*

3.4–b less_or_equal: Als gestandener Informatiker sollte man sich klar machen können, daß $m \leq n$ äquivalent zu $m - n \leq 0$ (bzw. $m \dot{-} n = 0$ für nicht–negative Zahlen) ist. Damit kann man die Definition für less_or_equal auch so hinschreiben:

$$\mathsf{less\_or\_equal}\ \overline{m}\ \overline{n} \equiv \begin{cases} \mathsf{T}, & \text{falls } m \dot{-} n = 0 \\ \mathsf{F}, & \text{sonst} \end{cases}$$

An dieser Stelle sollte es beim geneigten Leser bereits klingeln: das hierfür notwendige Werkzeug haben wir nämlich längst zur Hand. Es ist dies die "zero"–Funktion in Kombination mit dem eben definierten "subtract". Damit gelangen wir also zu folgendem Resultat:

$$\mathsf{less\_or\_equal} \equiv \lambda \mathtt{m}.\,\lambda \mathtt{n}.\,\mathsf{zero}\ (\mathsf{subtract}\ \mathtt{m}\ \mathtt{n})$$

Hier darf wiederum den kompletten λ–Term aufschreiben, wem's Freude bereitet...

Es ist $m < n$ falls $m - n < 0$ bzw. falls $(m+1) - n \leq 0$. Da die Subtraktion niemals Werte unter Null annehmen kann, reicht es zu testen, ob $(m+1) - n = 0$ ist, wofür wir die Funktion zero *verwenden.* less $\equiv \lambda m.\lambda n.$ zero (sub (s m) n)

3.4–c max: Hier suggeriert die Definition die Anwendung eines Conditionals gemäß folgender Formulierung:

"**Wenn** $m \leq n$ gilt, **dann** ist max $\overline{m}\ \overline{n}$ gleich $\overline{n}$, **ansonsten** ist max $\overline{m}\ \overline{n}$ gleich $\overline{m}$."

Damit gelangen wir zu folgendem Term:

$$\begin{aligned} \mathsf{max} &\equiv \lambda \mathtt{m}.\,\lambda \mathtt{n}.\,\mathsf{if\ less\_or\_equal}\ \mathtt{m}\ \mathtt{n}\ \mathsf{then}\ \mathtt{n}\ \mathsf{else}\ \mathtt{m} \\ &\equiv \lambda \mathtt{m}.\,\lambda \mathtt{n}.\,\mathsf{cond}(\mathsf{less\_or\_equal}\ \mathtt{m}\ \mathtt{n};\mathtt{n};\mathtt{m}) \\ &\equiv \lambda \mathtt{m}.\,\lambda \mathtt{n}.\,(\mathsf{less\_or\_equal}\ \mathtt{m}\ \mathtt{n})\ \mathtt{n}\ \mathtt{m} \end{aligned}$$

Insgesamt sollte man hierbei erkannt haben, wie wichtig die definitorische Erweiterung ist, wenn man vernünftig mit dem λ–Kalkül arbeiten will. Wer das (immer noch) nicht glaubt, der kann ja mal versuchen, dieselbe Herleitung mit den »nackten« λ–Termen durchzuexerzieren...

Lösung 3.5

3.5–a Rekursionsgleichungen für $f \equiv \mathsf{PRs}[b,h]$

$\vdash f\ \overline{0} = b$	BY unfold
$\vdash (\lambda \mathtt{n}.\ \ \mathtt{n}\ h\ b)\ \overline{0} = b$	BY reduce
$\vdash \overline{0}\ h\ b = b$	BY unfold
$\vdash (\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{x})\ h\ b = b$	BY reduce THEN reduce THEN reflexivity

$\vdash f\ (s\ n) = h\ (f\ n)$	BY unfold THEN reduce
$\vdash (s\ n)\ h\ b = h\ (f\ n)$	BY unfold
$\vdash (\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\ (n\ \mathtt{f}\ \mathtt{x}))\ h\ b = h\ (f\ n)$	BY reduce THEN reduce
$\vdash h\ (n\ h\ b) = h\ (f\ n)$	BY symmetry THEN unfold
$\vdash h\ ((\lambda \mathtt{n}.\ \ \mathtt{n}\ h\ b)\ n) = h\ (n\ h\ b)$	BY reduce THEN reflexivity

3.5–b $\mathsf{p}\,\overline{n} = \overline{m} \quad \equiv \quad \overline{n} = \mathsf{s}\,\overline{m}$:

Dieser Beweis ist ein wenig umfangreicher. Er wird in zwei Teile aufgespaltet:

1. Der Hauptbeweis
2. Ein Lemma über die Reduktion eines Teilredex'

Wir überlegen zunächst, wie die Reduktion von $\mathsf{p}\,\overline{n}$ unabhägig von einem konkreten n prinzipiell aussieht.

Vorüberlegung:

$$
\begin{aligned}
\mathsf{p}\ \overline{n} \quad &\equiv \quad (\lambda \mathtt{n}.\ (\mathtt{n}\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle).2)\ \overline{n} \\
&\longrightarrow \quad (\overline{n}\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle).2 \\
&\equiv \quad (\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^n\ \mathtt{x})\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle\ ).2 \\
&\longrightarrow \quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^n\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle\ ).2
\end{aligned}
$$

Nun können wir zwei Fälle für diesen Redex unterscheiden:

Erster Fall — $n = 0$**.** Dann gilt:

$$
\begin{aligned}
\mathsf{p}\ \overline{0} \quad &\equiv \quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^0\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle\ ).2 \\
&\equiv \quad \langle \lambda \mathtt{z}\ \overline{0},\ \overline{0}\rangle.2 \\
&\longrightarrow \quad \overline{0}
\end{aligned}
$$

Zweiter Fall — $n = m + 1$**.** Dann gilt:

$$
\begin{aligned}
\mathsf{p}\ \overline{m+1} \quad &\equiv \quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^{m+1}\ \langle \lambda \mathtt{z}.\overline{0},\ \overline{0}\rangle\ ).2 \\
&\equiv \quad (\ \ (\ \lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^m \\
&\qquad\quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)\ \ \langle \lambda \mathtt{z}.\ \overline{0},\ \overline{0}\rangle\ )\ ).2 \\
&\longrightarrow \quad (\ \ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^m \\
&\qquad\quad \langle \mathtt{s},\ \text{match } \langle \lambda \mathtt{z}.\ \overline{0},\ \overline{0}\rangle \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle\ ).2 \\
&\longrightarrow \quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^m\ \langle \mathtt{s},\ (\lambda \mathtt{z}.\ \overline{0})\ \overline{0}\rangle\ ).2 \\
&\longrightarrow \quad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^m\ \langle \mathtt{s},\ \overline{0}\rangle\ ).2
\end{aligned}
$$

siehe Lemma (*)

$$
\begin{aligned}
&\longrightarrow \quad \langle \mathtt{s},\ \mathtt{s}^m\ \overline{0}\rangle.2 \\
&\longrightarrow \quad \mathtt{s}^m\ \overline{0} \\
&\longrightarrow \quad \overline{m}
\end{aligned}
$$

Induktionsbeweis für Lemma (*) —
$(\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^n\ \langle \mathtt{s},\ \overline{0}\rangle \equiv \langle \mathtt{s},\ \mathtt{s}^n\ \overline{0}\rangle$:

Anfang — $n = 0$:

$(\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^0\ \langle \mathtt{s},\ \overline{0}\rangle \equiv \langle \mathtt{s},\ \overline{0}\rangle \equiv \langle \mathtt{s},\ \mathtt{s}^0\ \overline{0}\rangle$ $\surd$

Schritt: es gelte $(\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^n\ \langle \mathtt{s},\ \overline{0}\rangle \equiv \langle \mathtt{s},\ \mathtt{s}^n\ \overline{0}\rangle$.

Dann folgt:

$$
\begin{array}{rl}
 & (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^{n+1}\quad \langle \mathtt{s},\overline{0}\rangle \\
\longrightarrow & \ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle) \\
 & \qquad (\ (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)^n\quad \langle \mathtt{s},\ \overline{0}\rangle\ ) \\
\text{wegen Annahme} & \\
\equiv & (\lambda \mathtt{z}.\ \langle \mathtt{s},\ \text{match } \mathtt{z} \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle)\quad \langle \mathtt{s},\ \mathtt{s}^n\ \overline{0}\rangle \\
\longrightarrow & \langle \mathtt{s},\ \text{match } \langle \mathtt{s},\ \mathtt{s}^n\ \overline{0}\rangle \text{ with } \langle \mathtt{f},\mathtt{x}\rangle \mapsto \mathtt{f}\ \mathtt{x}\rangle \\
\longrightarrow & \langle \mathtt{s},\ \mathtt{s}\ (\mathtt{s}^n\ \overline{0})\rangle \\
\equiv & \langle \mathtt{s},\ \mathtt{s}^{n+1}\ \overline{0}\rangle
\end{array}
$$

Es gibt einen kürzeren Beweis ... wenn ich Zeit finde, schreibe ich ihn auf